Fonctions d'évaluations

Comment utiliser l'interface et l'assistant ?

Pour utiliser un élément ou une fonction :

1. dans la zone de texte en haut, posistionnez le curseur à l'endroit ou vous voulez l'inclure
2. Double cliquer sur la fonction ou l'élément souhaité dans une des deux listes en dessous, la fonction ou l'élément apparait automatiquement en haut dans la zone de texte

Deux cas possibles :

- Soit vous avez une fonction comme split, qui affiche un point “.” devant, exemple: .split(“-”)
- Soit vous avez une fonction sans point devant, donc avec () à la fin

Dans le **premier cas**, le champs à impacter doit être placé devant le point qui précède la fonction donc :

```
@FLD19.split('-')
```

Dans le **second cas**, le champs à impacter sera entre parenthèses, comme un argument en programmation :

Dans ce second cas, la position du champs dépend ensuite de la fonction, pour plus d'informations, voir le sommaire de cette page et cherchez la fonction concernée.

Vous pouvez également dans la zone **Options de capture** > **Fonction**, appeler vos propres fonctions. **todo**

Fonctions disponibles

Concaténation d'éléments

```
@concat
```

Permet de concaténer les valeurs de chacun des paramètres qui lui sont fournis.

Il est également possible de spécifier un séparateur.

Elle vous permet de réaliser des opérations de concaténation directement dans votre champ de recherche variable sans passer par un second champ de type spécifique.

Options de capture

Deux syntaxes sont possibles.

Sans spécifier de séparateur:

```
@concat(valeur1,valeur2,...,valeurN)
```

En spécifiant un séparateur:

```
@concat('@sep:separateur',valeur1,valeur2,...,valeurN)
```

Où *separateur* peut être le caractère (ou chaîne de caractères) de votre choix à l'exception du caractère quote simple.

Chaque valeur peut être :

- Une chaîne de caractères encadrée par des quotes simples.
- Un numérique.
- @self : une référence au champ lui-même.
- @<groupname> : une référence à une capture dans le filtre de champ. (effectuée dans le filtre de champ).

Date d'échéance

```
@dateecheance(@fldX,jours)
```

une macro qui fait une chose bien pratique de façon simple.

Options de capture

@fldX La référence au champ date.

jours Le nombre de jours que l'on souhaite ajouter à la date

Concaténation de Date

@concatdate permet de concaténer les valeurs de chacun des paramètres qui lui sont fournis et en plus leur applique la macro @Smartdate.

```
@concatdate(valeur1,valeur2,...,valeurN)
```

ou

```
@concat('@sep:separateur',valeur1,valeur2,...,valeurN)
```

Pour plus de détails se référer à la macro @concat

Concaténation d'éléments et conversion en Date

@concatdate permet de concaténer les valeurs de chacun des paramètres qui lui sont fournis et en plus leur applique la macro **@Smartdate**.

```
@concatdate(valeur1,valeur2,...,valeurN)
```

ou

```
@concat('@sep:separateur',valeur1,valeur2,...,valeurN)
```

Pour plus de détails se référer à la macro @concat

Condition OU entre plusieurs champs

@or renvoi 1 (sous forme de chaîne de caractère) si les champs champ1 OU champ2 ont été retrouvés sur le document. Dans le cas contraire la chaîne vide sera renvoyée.

```
@or(champ1,champ2)
```

Où champ1 et champ2 sont des références de champs (@FLDX)

Condition évaluée

```
@condition(<expression>)
```

Où *expression* doit être une expression python valide. Elle peut contenir des références à des champs d'analyse COLD (**@FLD***xxx*). **@condition** évalue l'expression python fournie en paramètre. Elle retourne 1 si l'expression est évaluée à vraie, et 0 si l'expression est évaluée à faux.

Avant évaluation les références **@FLD***xxx* sont remplacées par leurs valeurs et sont encadrées par des simples quotes (sauf pour les valeurs 0, 1, True, False). Attention donc car un nombre devient une chaîne de caractère ! Il peut être alors nécessaire d'utiliser des fonctions de conversion (long(), int(), float())

Conversion du jour numérique en texte

@int2day est une macro qui traduit un jour exprimé sous forme numérique dans sa représentation littérale.

```
@int2day(nombre)
```

Où nombre peut prendre les valeurs suivantes: * Une chaîne de caractères encadrée par des quotes simples.

- Un numérique.
- @self : une référence au champ lui-même.
- @<groupname> : une référence à une capture dans le filtre de champ.

```
@int2day(3) => mercredi
@int2day("7") => dimanche
```

En utilisant @self, si celui-ci vaut 5:

```
@int2day(@self) => vendredi
```

En utilisant une variable de capture <*jour*> qui vaut 1

```
@int2day(@<jour>) => lundi
```

Conversion du mois texte en mois numérique

```
@monthint(mois)
```

Où le paramètre *mois* est une chaîne de caractères qui représente l'écriture d'un mois sous sa forme littérale.

@monthint retourne le numéro (sur deux chiffres) du mois correspondant.

Ci dessous le tableau des correspondances entre un numéro et une représentation littérale du mois.

Numéro de mois en sortie	Valeurs possibles en entrée
01	ja, jan, janv, janu, janvi, janua, janvie, janvier, januar, january
02	f, fe, fev, feb, fevr, febr, fevri, febru, fevrie, februa, fevrier, februar, february
03	mar, mars, marc, march, marz
04	av, ap, avr, apr, avri, apri, avril, april
05	mai, may
06	juin, jun, june, juni
07	jul, july, juil, juill, juille, juillet
08	au, ao, aug, aou, aout, augu, augus, august
09	s, se, sept, septe, septem, septemb, septembe, septembr, septembre, september
10	o, oc, oct, octo, octob, octobe, octobr, october, octobre, okt, oktober
11	n, no, nov, nove, novem, novemb, novembr, novembe, novembre, november
12	d, de, dec, dece, decem, decemb, decembr, decembe, decembre, december, dez, dezember

Conversion du mois numérique en texte

@int2month est une macro qui traduit un mois exprimé sous forme numérique dans sa représentation littérale.

Valeurs alternatives

@alternative renvoi la première valeur non vide parmi une liste de valeurs possibles.

```
@alternative(@FLD4,@FLD3,'Non renseignée')
```

La macro **@alternative** prend au minimum 2 paramètres.

Chaque paramètre alternatif peut être :

- Une chaîne de caractère (encadrée ou non par des simples quotes ou des doubles quotes)
- Une référence à un champ (**@FLD***xxx*).

Exemple

ID	Description du champ	Valeur retrouvée
3	Adresse de facturation	2 Rue Pégoud, 90130 PETIT-CROIX
4	Recherche Adresse de livraison	1 Rue du Général de Gaulle, 90130 Montreux-Château

Nous avons deux champs. L'un nous retrouve l'adresse de facturation. Le second l'adresse de livraison. Nous voudrions, si l'adresse de livraison n'est pas spécifiée, choisir l'adresse de facturation.

Créons un champ supplémentaire de type spécifique et utilisons la macro @alternative

```
@alternative(@FLD4,@FLD3)
```

Le premier choix se porte sur le champ 4 (Recherche Adresse de livraison). S'il n'est pas trouvé on prendra la valeur du champ 3 (Adresse de facturation).

Et si aucun n'est trouvé et que l'on souhaite indexer une valeur par défaut :

```
@alternative(@FLD4,@FLD3,'Non renseignée')
```

Convertir en entier (cast)

Double cliquez sur la fonction voulue donc “**Convertir en entier**”, cela fait apparaitre la fonction dans la zone texte au dessus.

La fonction **long()** prend en paramètre une variable qu'elle transforme en entier.

Dans la liste de champs en-dessous, il suffit donc de choisir le champs qui contient la valeur que vous souhaitez convertir. Faites un double clic.

Vous verrez alors apparaitre la pseudo-variable, correspondant au champs, dans la zone texte, en argument de la fonction. A moins d'une erreur, il n'y a plus qu'à cliquer sur Valider pour que votre choix soit prêt à être appliqué. Il est indiqué dans le formulaire, section “**Options de capture**”

Couper sur séparateur (split)

La fonction split génère un tableau contenant toutes les parties.

Exemple :
“559_1003”.split(“_”) donne

[“559”, “1003”]

Ou 559 est la première case donc index 0 : “559_1003”.split(“_”)[0]
Et 1003 est la deuxième case donc index 1 : “559_1003”.split(“_”)[1]

Vous pouvez aussi parcourir les cases depuis la fin donc en partant de -1 pour la première case :

“559_1003”.split(“_”)[-1] donne “1003”

Extraction d'une partie de la chaine

Cette macro permet de ne prendre qu'une partie d'une valeur.

Il faut savoir qu'une chaine de texte en python est en réalité un tableau de caractères affichés les uns à la suite des autres.

C'est à dire : “glou glou”
est en fait un tableau qui contient : [“g”,“l”,“o”,“u”,“g”,“l”,“o”,“u”]

Donc utiliser la macro ainsi :

@champ[0] affichera “g” uniquement

Ou encore :

@champ[1:3] affichera “lou”

Rechercher la position d'une chaine xxx depuis le debut

@champ.find('z')

La macro parcoure la chaine en affectant un numéro à chaque caractère (espaces compris), depuis 0. Dès qu'elle trouve une première fois la valeur xxx, elle arrête la détection et renvoie la position du premier caractère.

Si vous souhaitez obtenir la position du dernier caractère, vous pouvez utiliser la macro inverse rfind.

Exemple :

si le champ contient “le cheval s'appelle Miguel” et que vous cherchez le premier a, utilisez @champ.find('a') Vous obtiendrez 7

La macro compte les espaces comme des caractères.

From:
https://wiki.ezdev.fr/ - **EzGED Wiki**

Permanent link:
https://wiki.ezdev.fr/doku.php?id=doc:v3:acquisition:apprentissage:fonctionsevaluation&rev=1528294895

Last update: **2023/03/17 09:56**